

PARTE II

ESCUELA DE MICROSERVICIOS



ÍNDICE DE CONTENIDOS

TRATAMIENTO DE FICHEROS
APIS REST

ANGULAR MATERIAL

PAGINACIÓN

FORMULARIOS PLANTILLA

FONTAWESOME

COMUNICACIÓN ENTRE
COMPONENTES

TESTING CON KARMA &
JASMINE

PRODUCCIÓN ANGULAR



ÍNDICE DE CONTENIDOS

SPRING SECURITY

JWT

RESILIENCIA

JENKINS

NEXUS

DOCKER

DOCKER COMPOSE

CICLO COMPLETO CI/CD



TRATAMIENTO FICHEROS API REST

EN LAS APIS REST, EL
FORMATO DE INTERCAMBIO
PREDEFINIDO ENTRE EL
CLIENTE Y EL SERVIDOR ES
APPLICATION/JSON

AHORA BIEN, ¿CÓMO
AFECTA ESTO SI NECESITO
INCLUIR FICHEROS
BINARIOS EN EL API?

PROBLEMA: AUMENTO
TAMAÑO DE LAS
PETICIONES

TRATAMIENTO FICHEROS API REST

@LOB MAPEAMOS EL ATRIBUTO DE LA ENTIDAD COMO FICHERO EN LA BASE DE DATOS

@JSONIGNORE: EL CONTENIDO DEL FICHERO NO SE SERIALIZA

EN SU LUGAR INTRODUCIMOS UNA BANDERA O FLAG, QUE PERMITA DISTINGUIR AL CLIENTE LA EXISTENCIA O NO DEL FICHERO ADJUNTO ASOCIADO AL REGISTRO

TRATAMIENTO FICHEROS API REST

**SOLUCIÓN: TRATAMIENTO INDEPENDIENTE Y
CONCURRENTE**

**CREAMOS MÉTODOS DE DESCARGA DE FICHERO
SEPARADOS DEL GET**

**DUPLICAMOS LOS POST Y PUT PARA QUE SE PUEDAN
SUBIR FICHEROS ASOCIADOS A UN REGISTRO
MULTIPART/FILE**

TRATAMIENTO FICHEROS

API REST

OTRAS ALTERNATIVAS: GUARDAR LA RUTA Y NO EL FICHERO EN LA BASE DE DATOS

FORMATEAR EL FICHERO EN BASE64 Y ADJUNTARLO

DEFINIR UN TAMAÑO LÍMITE DE FICHERO EN EL SERVIDOR



ANGULAR MATERIAL

LA FILOSOFÍA DEL
DESARROLLO POR
COMPONENTES, ES
LLEVADA A CABO POR LA
CASA ANGULAR

ANGULAR MATERIAL NOS
OFRECE MULTITUD DE
COMPONENTES LISTOS
PARA SU USO

DEBO INTEGRAR ESTA
LIBRERÍA EN MI PROYECTO
VIA NPM

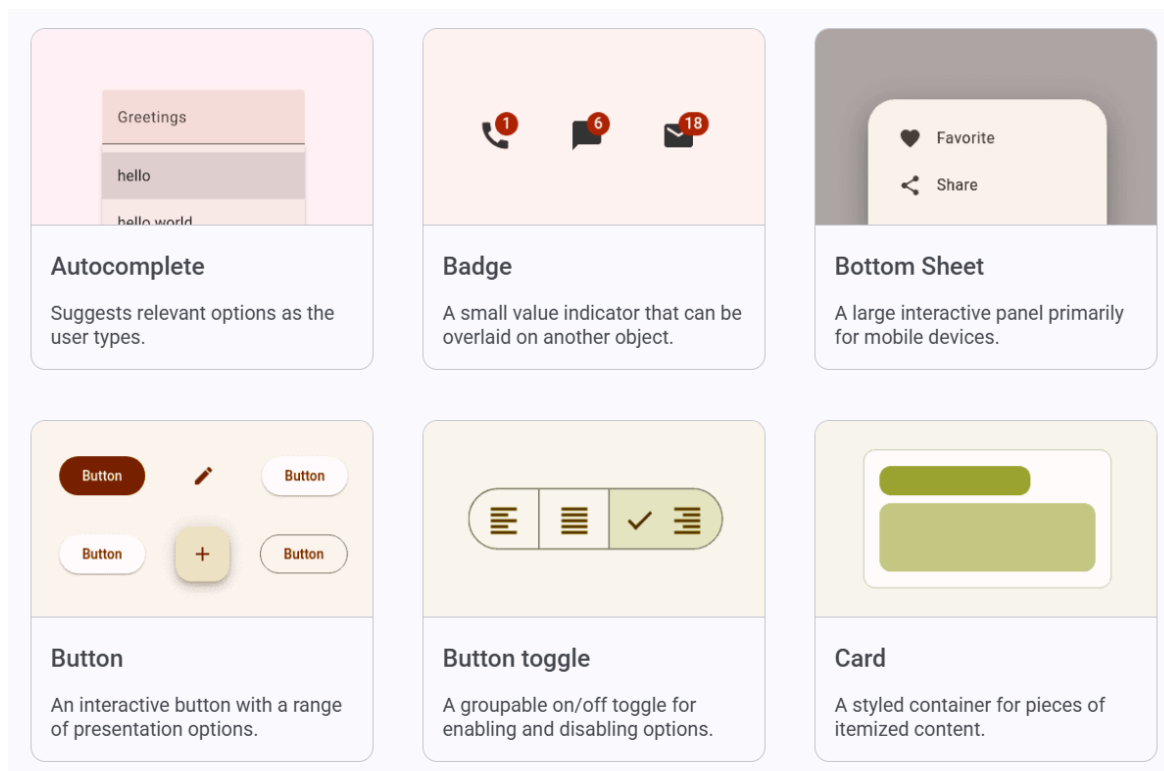
ANGULAR MATERIAL

**INSTALAMOS DESDE EL REPOSITORIO NPM CON
NG ADD @ANGULAR/MATERIAL**

CONFIGURAMOS LAS OPCIONES POR DEFECTO

ANGULAR MATERIAL

HAY MUCHOS COMPONENTES DISPONIBLES

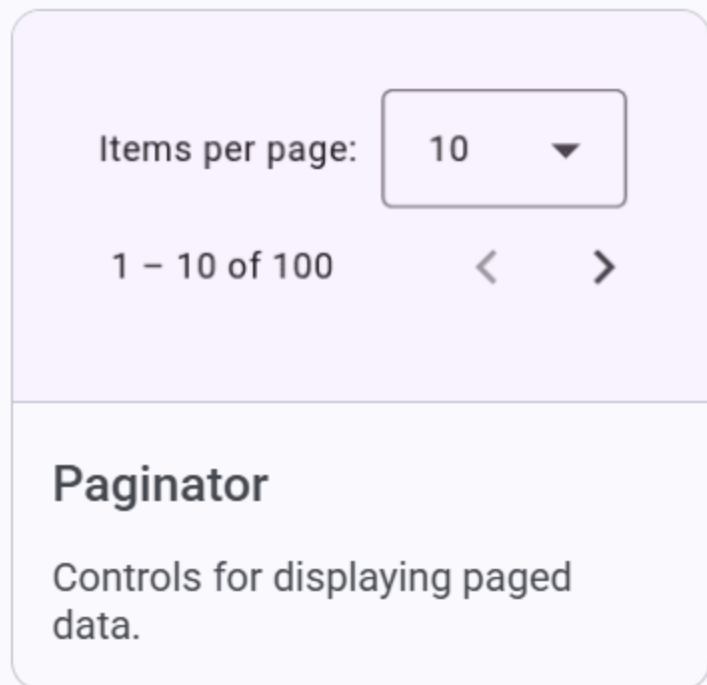


ANGULAR MATERIAL

PARA PODER UTILIZARLOS, HAY QUE LEER SU DOCUMENTACIÓN

CADA COMPONENTE, SUELE ESTAR ENCAPSULADO EN UN MÓDULO, QUE A SU VEZ TIENE ASOCIADA SUS DIRECTIVAS, SERVICIOS Y CLASES PROPIAS

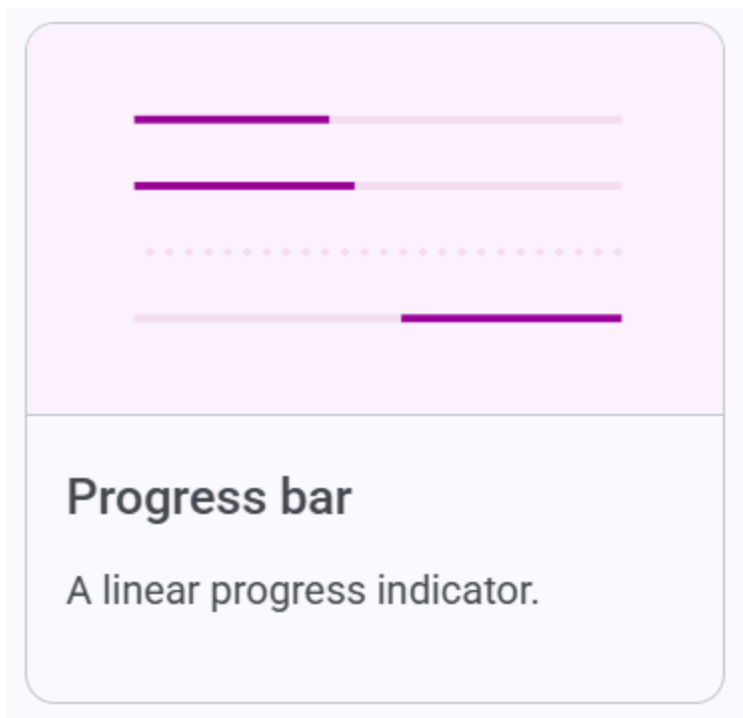
ANGULAR MATERIAL



**IDEAL PARA PAGINAR
RESULTADOS**

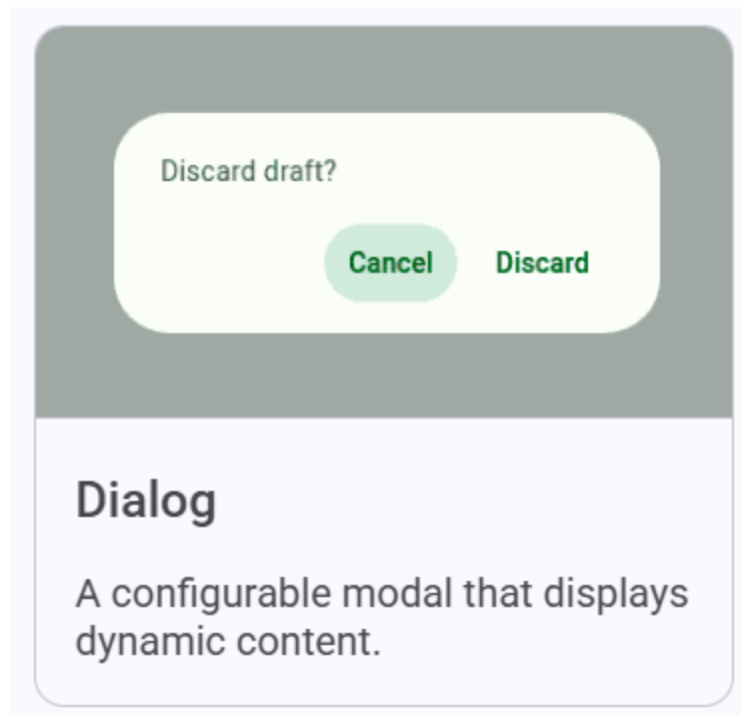
**LLEVA ASOCIADA UNA
CONFIGURACIÓN EN EL
SERVIDOR**

ANGULAR MATERIAL



**UN INDICADOR SIMPLE,
MIENTRAS SE ESPERA A
QUE SE COMPLETEN
OPERACIONES EN EL
CLIENTE**

ANGULAR MATERIAL



**PERMITE LA CREACIÓN
DE VENTANAS POP, CON
DIÁLOGOS O DETALLES
INFORMATIVOS**



FORMULARIOS PLANTILLA

POR SER LOS FORMULARIOS
UN ELEMENTO HABITUAL
EN LAS PÁGINAS WEB,
TIENEN UN SOPORTE
ESPECIAL EN ANGULAR

SU MODELADO PERMITE
DEFINIR EL FORMULARIO Y
LOS ELEMENTOS QUE LO
COMPONEN, GESTIONANDO
SU VALIDACIÓN DE
MANERA DECLARATIVA

FORMULARIOS PLANTILLA

**CADA ELEMENTO DEL FORMULARIO QUEDA
ASOCIADO CON LA DIRECTIVA NGMODEL**

**EL FORMULARIO EN SU CONJUNTO, SE DEFINE CON
NGFORM**

FORMULARIOS PLANTILLA

**SOBRE CADA ELEMENTO, PUEDO EVALUAR SU
CONDICIÓN DE TOCADO, MODIFICADO Y VÁLIDO**

**SOBRE EL FORMULARIO EN CONJUNTO PUEDO
COMPROBAR SI SU ESTADO ES VÁLIDO O HA SIDO
ENVIADO**

Y TODO DE FORMA DECLARATIVA



FONTAWESOME

LIBRERÍA DE ICONOS
VECTORIALES

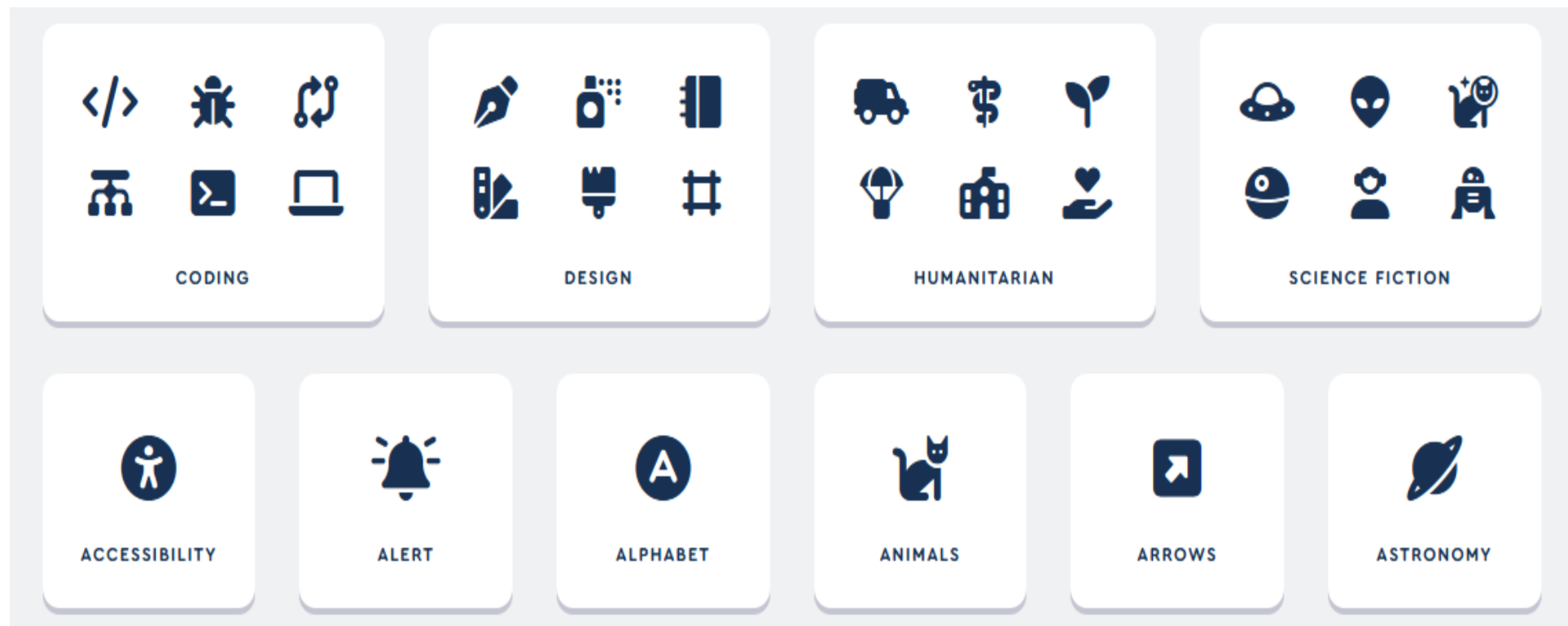
ACCESIBLE POR NPM Y CON
CIENTOS DE ICONOS
LISTOS PARA USO

FONTAWESOME

**ES UNA LIBRERÍA DE CÓDIGO ABIERTO, CON
MULTITUD DE ICONOS DISPONIBLES PARA
PERSONALIZAR Y ENRIQUECER NUESTRAS VISTAS**

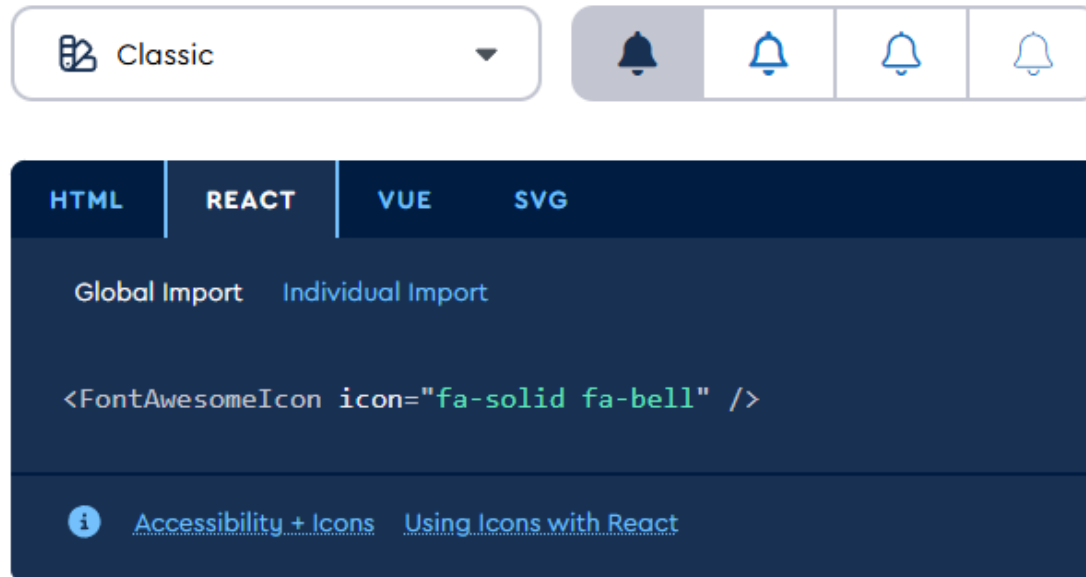
**INTERNAMENTE, ESTÁN CODIFICADOS COMO
GRÁFICOS VECTORIALES, POR LO QUE SU USO ES
ÓPTIMO EN EL PESO DE LOS ARCHIVOS Y EL
ESCALADO**

FONTAWESOME



FONTAWESOME

CADA ICONO, TIENE UN NOMBRE QUE LO IDENTIFICA
EN REALIDAD, EL ICONO, ES UN COMPONENTE





COMUNICACIÓN ENTRE COMPONENTES

CUANDO NECESITO
COMUNICAR INFORMACIÓN
ENTRE PÁGINAS, PUEDO
SEGUIR VARIAS
ESTRATEGIAS

UN CASO DISTINTO, ES
CUANDO TENGO UNA
PÁGINA FORMADA POR
DISTINTOS
COMPONENTES, ENTRE LOS
QUE TAMBIÉN PUEDO
COMUNICARME

COMUNICACIÓN ENTRE COMPONENTES

**PARA PASAR INFORMACIÓN ENTRE PÁGINAS, TENGO
VARIAS ALTERNATIVAS:**

- **URL**
- **VÍA SERVICIO COMÚN**
- **USANDO EL LOCALSTORAGE**

COMUNICACIÓN ENTRE COMPONENTES

SI QUIERO PASAR INFORMACIÓN ENTRE COMPONENTES CON RELACIÓN JERÁRQUICA, PUEDO:

- **ACCEDER A UN HIJO DESDE EL PADRE - VIEWCHILD**
- **PASAR INFO DEL HIJO AL PADRE - OUTPUT**
- **PASAR INFO DEL PADRE AL HIJO - INPUT**
- **PASAR INFO ENTRE "HERMANOS" - SERVICE**

COMUNICACIÓN ENTRE COMPONENTES

**A PARTIR DE LA VERSIÓN 18, SE INTRODUCEN
ADEMÁS LAS FUNCIONES**

- **INPUT**
- **OUTPUT**
- **MODEL**


COMUNICACIÓN ENTRE COMPONENTES

ESTAS NUEVAS FUNCIONES, SIRVEN PARA CONTROLAR EL ESTADO DE LOS COMPONENTES, ES DECIR, EL VALOR DE SUS ATRIBUTOS EN UN MOMENTO DADO Y LA REACCIÓN ANTE ESOS CAMBIOS.

Y SON UN SUBTIPO DE SIGNAL, EL NUEVO CONCEPTO TRAÍDO EN ANGULAR 17, DISEÑADO PARA MANTENER EL CONTROL SOBRE EL ESTADO DE LA APLICACIÓN, EN SUSTITUCIÓN DE ZONEJS



TESTING CON KARMA Y JASMINE



EL ENTORNO DEL CLIENTE
TAMBIÉN SE PRESTA A LA
FASE DE TESTING CON LA
COLABORACIÓN DE ESTAS
DOS LIBRERÍAS

KARMA Y JASMINE, SON
DEPENDENCIAS POR
DEFECTO DE TODO
PROYECTO ANGULAR

PODEMOS PLANEAR TEST
FUNCIONALES, NO
FUNCIONALES, UNITARIOS
E INTEGRALES

TESTING CON KARMA Y JASMINE

**LOS FICHEROS CON EXTENSIÓN SPEC.TS, SE CREAN
POR DEFECTO PARA CADA COMPONENTE, CLASE O
DIRECTIVA CREADOS.**

**EN ELLOS, SE DESARROLLAN LOS CASOS DE TEST
RELATIVOS A ESE FICHERO DE LA APLICACIÓN**

TESTING CON KARMA Y JASMINE

**LA LIBREIA JASMINE, ES LA QUE EN REALIDAD
PROGRAMA CADA CASO DE PRUEBA O "IT" ITEM
TEST, IDENTIFICADO CON UNA BREVE DESCRIPCIÓN**

**POSTERIORMENTE, ES KARMA QUIEN LANZA UN
PROCESO WEB, EVALUANDO TODAS Y CADA UNA DE
LOS CASOS DE PRUEBA DEFINIDOS EN EL PROYECTO,
SI BIEN SE PUEDE SER MÁS SELECTIVO,
EJECUTANDO SÓLO ALGUN FICHERO TEST**

TESTING CON KARMA Y JASMINE

CLASES Y MÉTODOS CLAVE

TESTBED

CONFIGURETESTMODULE()

COMPONENTFIXTURE

DONE()

NATIVEELEMENT

COMPONENTINSTANCE

DETECTCHANGES()



PUESTA EN PRODUCCIÓN ANGULAR

AL COMPILAR UNA
APLICACIÓN ANGULAR
TODO QUEDA TRADUCIDO A
CSS, JS Y HTML

HAY VARIAS OPCIONES
PARA DESPLEGAR LA
APLICACIÓN

HAY MATICES IMPORTANTES
SI LO DESPLEGAMOS
CONJUNTO A NUESTRO
SERVIDOR DE BACK O NO

PUESTA EN PRODUCCIÓN

EL SIMPLE COMANDO `NG BUILD`, NOS GENERA LA SUBCARPETA `DIST`, DONDE SE GENERAN LOS `CSS`, `JS` Y `HTML` RESULTANTES

HAY DOS VERSIONES DE RENDERIZADO, EL `CSR` O `CLIENTE SIDE RENDERING` (TRADICIONAL) Y EL `SSR` O `SERVER SIDE RENDERING` (MÁS NOVEDOSO, OPTIMIZADO PARA `SEO`)

PUESTA EN PRODUCCIÓN

LAS SUBCARPETAS BROWSER Y SERVER, SON RESPECTIVAMENTE EL RESULTADO DE LAS COMPILACIONES CSR Y SSR

AMBAS DISTRIBUCIONES, SON CANDIDATAS DE DESPLEGAR EN UN SERVIDOR PÚBLICO DONDE SE SIRVE NUESTRA APP ANGULAR. PARA SSR, HACE FALTA NODE.JS EN EL SERVIDOR.



SPRING SECURITY

EL MÓDULO DE SEGURIDAD
ES UNA LIBRERÍA
INDEPENDIENTE EN SPRING

PODEMOS GESTIONAR LA
AUTENTICACIÓN DE
USUARIOS Y SU
AUTORIZACIÓN A USAR
DETERMINADAS APIS EN
FUNCIÓN DE SU ROL

PODEMOS OPTAR ENTRE
VARIOS MODELOS DE
AUTENTICACIÓN

SPRING SECURITY CONCEPTOS

AUTENTICACIÓN: PROCESO POR EL QUE IDENTIFICAMOS A UN USUARIO EN EL SERVIDOR, NORMALMENTE, TRAS CONTRASTAR SUS CREDENCIALES CON LAS ALMACENADAS EN EL SISTEMA

AUTORIZACIÓN: UNA VEZ PRODUCIDA LA AUTENTICACIÓN, UN USUARIO PUEDE ESTAR AUTORIZADO O NO A DETERMINADA OPERACIÓN EN FUNCIÓN DE SU ROL O PERFIL

SPRING SECURITY CONCEPTOS

**FILTRO: PROCESO QUE SE EJECUTA ANTES Y
DESPUÉS DEL CONTROLLER AL QUE APLICA. FORMA
PARTE DEL ESTÁNDAR DE JEE**

SPRING SECURITY

MÉTODOS

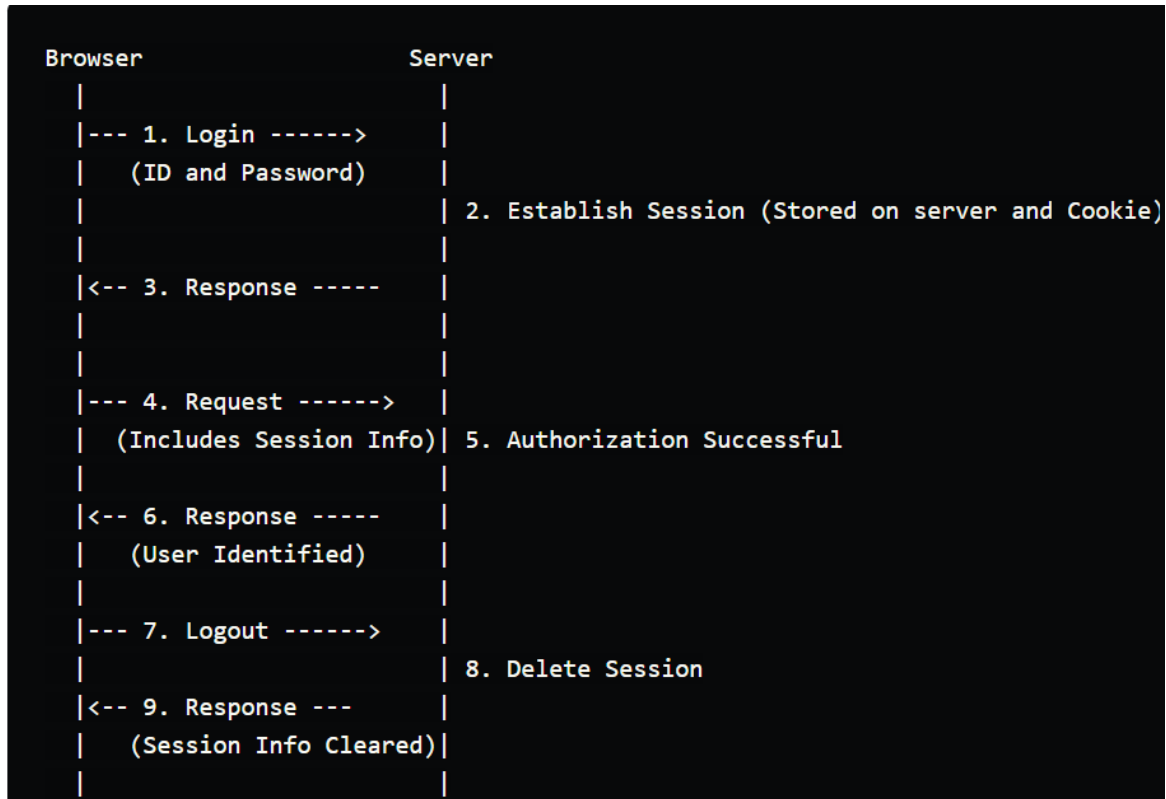
**SESIÓN: TRADICIONAL. SESIÓN ALMACENADA EN
SERVIDOR. SÓLO PARA UN SERVIDOR. CON ESTADO**

**JWT: APIS DISTRIBUIDAS. SIN ALMACENAMIENTO EN
EL SERVIDOR. PARA VARIOS SERVIDORES. SIN
ESTADO**

OAUTH: REDES SOCIALES, CUENTAS PREEXISTENTES

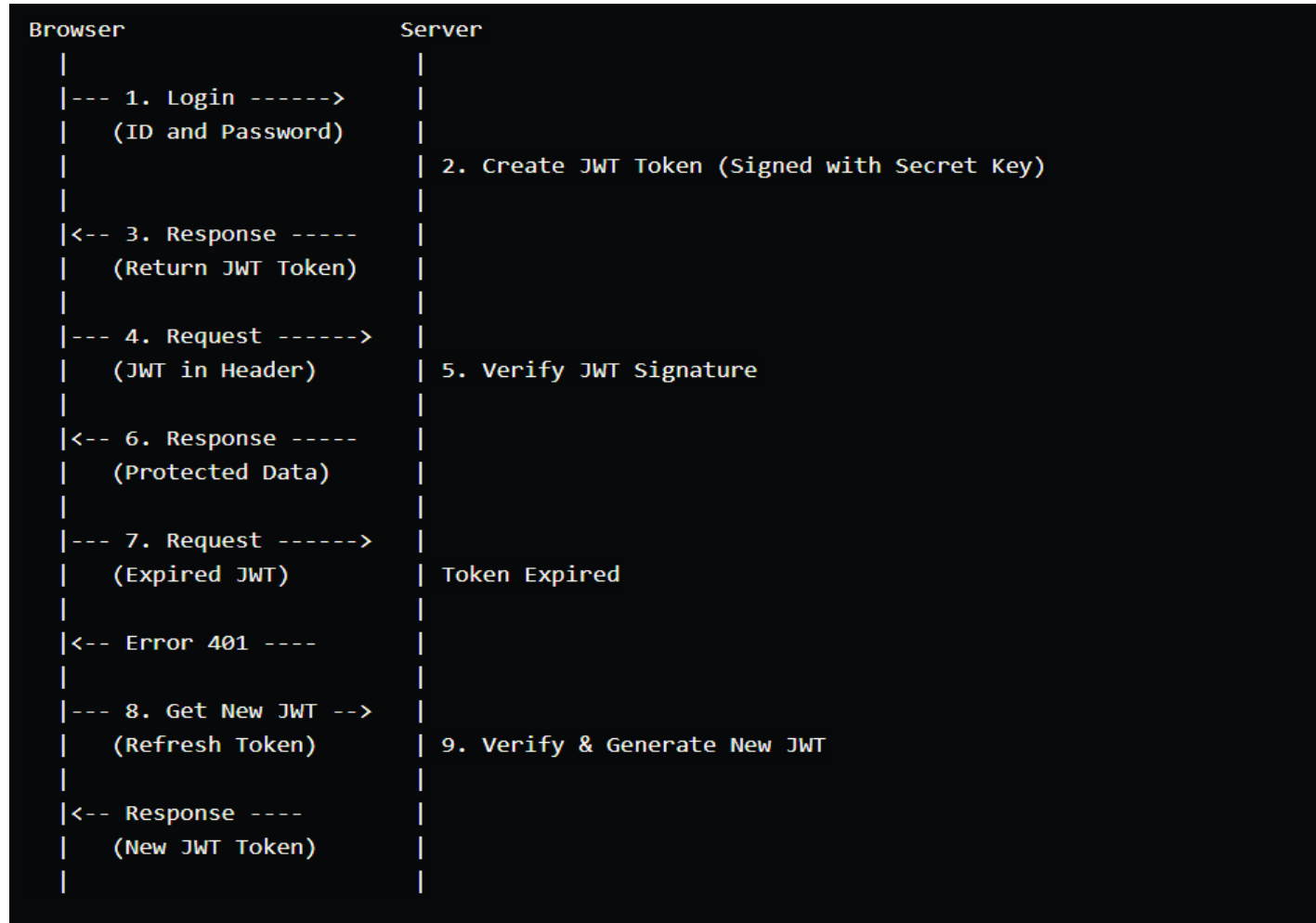
SPRING SECURITY

SESIÓN



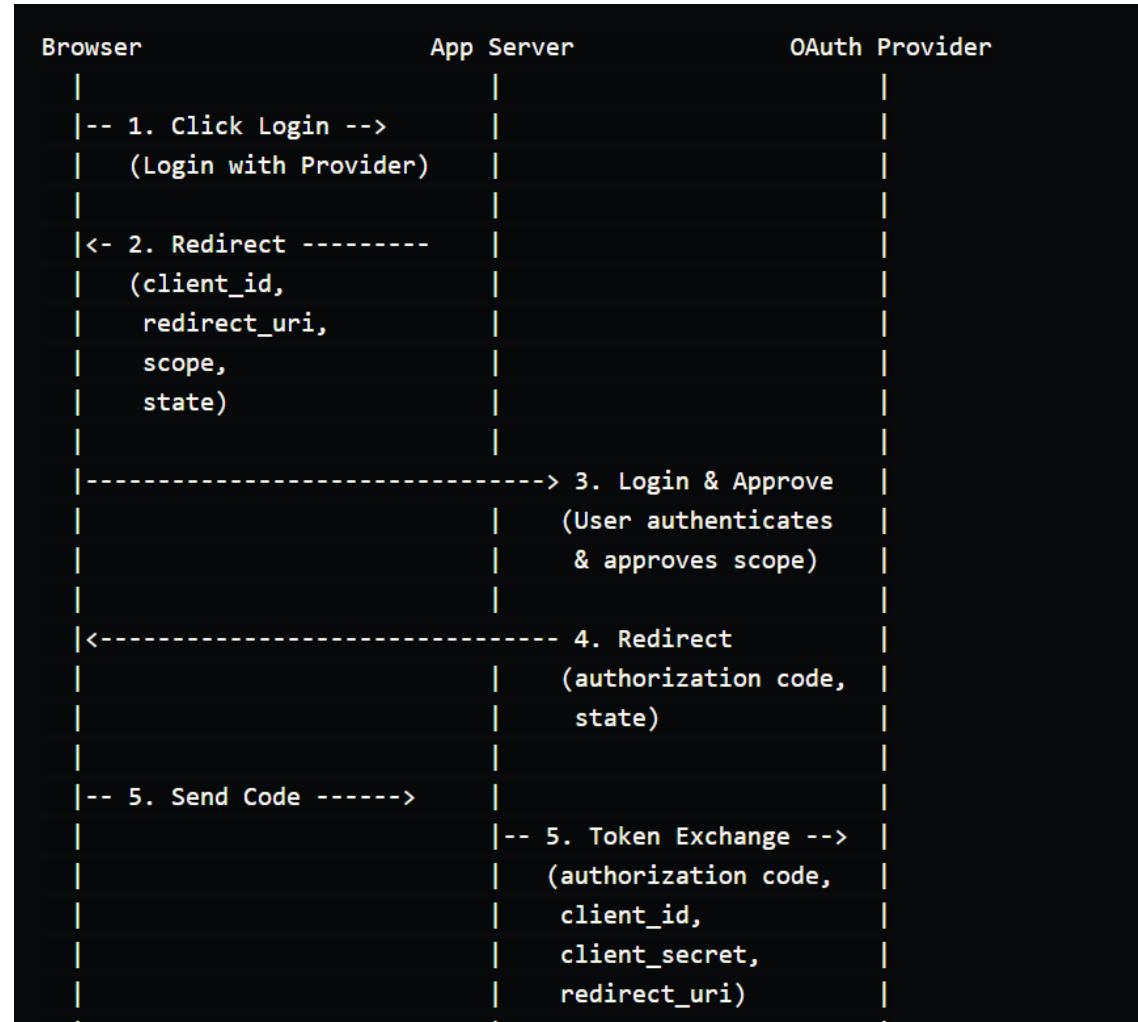
SPRING SECURITY

JWT



SPRING SECURITY

OAUTH



A 10x10 grid of dots. The rightmost column of dots is highlighted with a vertical orange bar.

```
|  
|  
|<- 6. Tokens -----  
|    (access token,  
|      refresh token,  
|      expiration time,  
|      id_token)  
|  
|<- 7. Create Session --  
|    (Store tokens &  
|      create user session)  
|  
|-- 8. Request with ----  
|    Expired Token  
|  
|  
|-- 8. Refresh Token -->  
|    (refresh token)  
|  
|<- 8. New Token -----  
|    (new access token)  
|  
|-- 9. Logout ----->  
|  
|<- 9. Clear Session ---  
|    (& optional redirect  
|      to provider logout)
```

SPRING SECURITY

CLASES PRINCIPALES

@CONFIGURATION, @ENABLEWEBSECURITY

**LA CLASE CON ESTAS ANOTACIONES, GESTIONA
TODA LA CONFIGURACIÓN DE SEGURIDAD**

PASSWORDENCODER

**LA CLASE EMPLEADA EN LA CODIFICACIÓN DE
CONTRASEÑAS**

SPRING SECURITY

CLASES PRINCIPALES

CREDENCIALES

**EL BEAN QUE REPRESENTA LOS DATOS RECIBIDOS
DEL USUARIO EN EL CONTROLLER**

USERDETAILS

**LA CLASE QUE PROPORCIONA A SPRING LA
INFORMACIÓN DEL USUARIO DE MODO
ESTANDARIZADO**

SPRING SECURITY

CLASES PRINCIPALES

USERDETAILSERVICE

LA CLASE QUE PERMITE ACCEDER A LA INFORMACIÓN DEL USUARIO Y COMPONER SU USERDETAIL

USERNAMEPASSWORDAUTHENTICATIONTOKEN

EL USUARIO Y CONTRASEÑA QUE REPRESENTAN LAS CREDENCIALES DEL EN EL CONTEXTO DE SEGURIDAD DE SPRING

SPRING SECURITY

CLASES PRINCIPALES

USERNAMEPASSWORDAUTHENTICATIONFILTER

**EL FILTRO QUE COMPRUEBA LA EXISTENCIA DE LAS
CREDENCIALES DEL USUARIO EN EL SISTEMA**

**FILTRO AUTORIZACIÓN: COMPRUEBA LA CABECERA Y
OBTIENE LA AUTORIZACIÓN DE LA PETICIÓN**



RESILIENCIA

LOS MICROSERVICIOS
FALLAN, PERO PODEMOS
APLCIAR UNA
POLÍTICA QUE SUPERVISE
EL RENDIMIENTO Y
MITIGUE LA EXPANSIÓN DE
ERRORES

RESILIENCE4J ES UNA
IMPLEMENTACIÓN
COMPLETA DE DISTINTOS
PATRONES, ACCESIBLE
DESDE EL STARTER SPRING
CLOUD CIRCUIT BREAKER



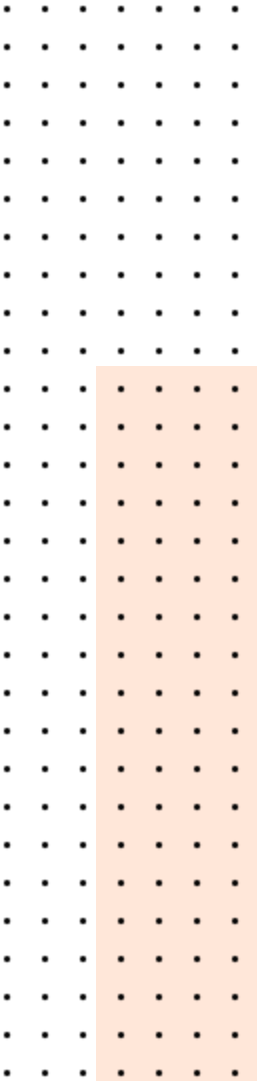
RESILIENCIA PATRONES

CIRCUIT BREAKER

RATE LIMITER

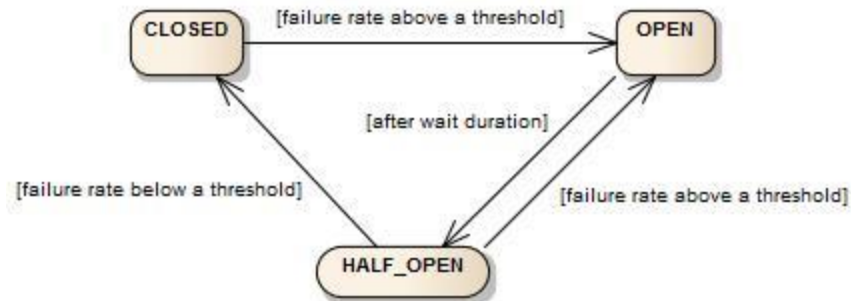
RETRY

TIMELIMITER



RESILIENCIA PATRONES

CIRCUIT BREAKER



PARAMETRIZAMOS EL SISTEMA PARA QUE ANTE UNA TASA DE FALLOS DETERMINADA, SE EJECUTEN MÉTODOS POR DEFECTO

RESILIENCIA PATRONES

CIRCUIT BREAKER

SE PUEDE COMBINAR CON SPRING ACTUATOR PARA MONITORIZAR CUANDO EL SERVICIO QUEDA ABIERTO (NO ADMITE LLAMADAS) CERRADO (FUNCIONA CORRECTAMENTE) Y SEMI-ABIERTO (UN ESTADO DE TRANSICIÓN)

RESILIENCIA PATRONES

RATE LIMITER

CON ESTE PATRÓN PODEMOS DEFINIR UNA TASA MÁXIMA DE SERVICIO, A PARTIR DE LA CUAL, EL SISTEMA HA ESTIMADO QUE NO PUEDE ATENDER MÁS PETICIONES Y DERIVAR A UN MÉTODO POR DEFECTO

RESILIENCIA PATRONES

RETRY

EL FALLO ES ALGO QUE CONSIDERA PROBABLE ESTE PATRÓN, POR LO QUE EL REINTENTO CONFIGURADO, SE LLEVA A CABO DE MANERA AUTOMÁTICA

TRAS SUPERARSE EL NÚMERO DE INTENTOS, SE EJECUTA EL MÉTODO DE FALLO PREVISTO DE MANERA OPCIONAL

RESILIENCIA PATRONES

TIMELIMITER

ESTE PATRÓN ESTIMA UNA DURACIÓN DETERMINADA EN LA EJECUCIÓN DE UN SERVICIO. SUPERADO ESE LÍMITE, SE EJECUTA EL MÉTODO POR DEFECTO PREVISTO



DOCKER

DOCKER INTEGRA TODO LO
NECESARIO PARA
EMPAQUETAR NUESTRO
SOFTWARE, CREANDO UNA
IMAGEN

UNA VEZ CREADA LA
IMAGEN, SE PUEDE
EJECUTAR UN
CONTENEDOR

EL CONTENEDOR SE PUEDE
EJECUTAR ALLÁ DONDE
HALLA UNA MÁQUINA CON
DOCKER

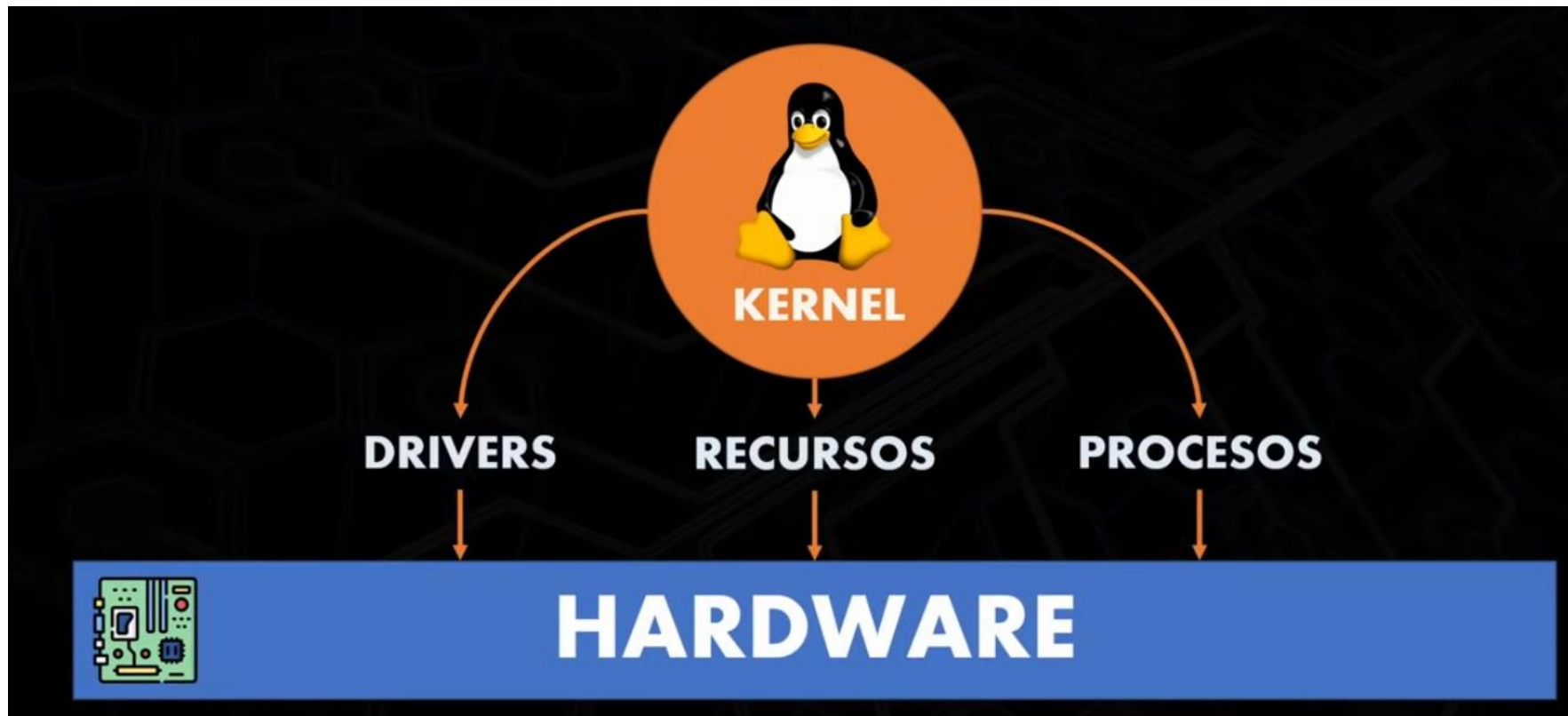
DOCKER VS MÁQUINAS VIRTUALES

**EN LAS MÁQUINAS VIRTUALES, LA CONFIGURACIÓN
REQUIERE DE UN SISTEMA OPERATIVO COMPLETO**

**LA MÁQUINA VIRTUAL VIRTUALIZA EL HARDWARE,
MIENTRAS QUE LOS CONTENEDORES VIRTUALIZAN A
TRAVÉS DEL SISTEMA OPERATIVO Y SON MÁS
LIGEROS**

**EL KERNEL ES COMÚN PARA DISTINTOS
CONTENEDORES. AHÍ RESIDE LA CLAVE**

DOCKER VS MÁQUINAS VIRTUALES



DOCKER VS MÁQUINAS VIRTUALES



DOCKER VS MÁQUINAS VIRTUALES



DOCKER VS MÁQUINAS VIRTUALES

LOS GRUPOS DE CONTROL Y LOS ESPACIOS DE NOMBRE SON LAS CARACTERÍSTICAS DEL KERNEL DE LINUX QUE NOS DAN EL AISLAMIENTO Y LA ASIGNACIÓN DE RECURSOS

DE MANERA QUE A TRAVÉS DE FUNCIONES INTERNAS DEL SISTEMA OPERATIVO, SE GESTIONAN RECURSOS HARDWARE EXCLUSIVOS PARA CADA CONTENEDOR, SIN NECESIDAD DE TENER EL SO COMPLETO

DOCKERFILE

**DOCKERFILE ES EL FICHERO QUE DESCRIBE CÓMO
COMPONER LA IMAGEN Y LA PLANTILLA DEL
PROYECTO**

**PARTIENDO DE UNA IMAGEN BASE (YA SEA UN SO O
UN ENTORNO COMO JDK) SE AGREGA EL PROYECTO
COMPILADO (JAR)**

**LA IMAGEN QUEDA LISTA PARA USARSE EN NUESTRO
ENTORNO**

DOCKERFILE

**EJEMPLO DE IMAGEN PARTIENDO DE LA ÚLTIMA
VERSIÓN DE JAVA, AGREGAMOS NUESTRA APP Y
ESPECIFICAMOS EL PUNTO DE ENTRADA**

FROM OPENJDK

ADD REST.JAR RESTCONT.JAR

ENTRYPOINT JAVA -JAR RESTCONT.JAR

DOCKER COMANDOS

DOCKER IMAGES => LISTADO DE IMÁGENES LOCAL

DOCKER IMAGES RM <IMAGEN>:<VERSIÓN> => LISTADO DE IMÁGENES LOCAL

DOCKER PS => CONTENEDORES EN EJECUCIÓN

DOCKER BUILD -T <IMAGEN> . => CONSTRUIMOS UNA IMAGEN

DOCKER RUN -P 8085:9456 <IMAGEN>


DOCKER PULL <NOMBRE_IMAGEN> => DESCARGA LA ÚLTIMA VERSIÓN

DOCKER TAG <IMAGEN> USUARIO/REPO:VERSION => CREAMOS UNA VERSIÓN

DOCKER PUSH USUARIO/REPO:VERSION => SUBIMOS VERSIÓN A DOCKERHUB



CICLO
COMPLETO
CI/CD



DE CÓMO GENERAR
VERSIONES DE UN
PROYECTO SIGUIENDO EL
FLUJO DE GIT FEATURES

PASOS CICLO COMPLETO

0 TENER ACTUALIZADO EL REPO LOCAL

GIT PULL

1 CREO Y SALTO A LA RAMA DE DESARROLLO

GIT CHECKOUT -B <RAMA>

2 MODIFICAMOS LA VERSIÓN EN POM

0.2.3-SNAPSHOT

3 TRABAJAMOS HASTA DAR POR BUENA LA MEJORA

GIT ADD .

GIT COMMIT -M <MENSAJE>

PASOS CICLO COMPLETO

4 ACTUALIZAMOS LA RAMA REMOTA DE DESARROLLO

GIT PUSH -U ORIGIN <RAMA>

5 CREAMOS LA PULL REQUEST EN GITHUB

ASIGNAMOS REVISORES OPCIONALES

6 LOS REVISORES RECIBIEN UN AVISO POR CORREO

DEBEN SER COLABORADORES PREVIAMENTE

7 EL REVISOR SE ACTULIZA SU REPOSITORIO

GIT PULL

PASOS CICLO COMPLETO

9 EL REVISOR SALTA A LA RAMA DE DESARROLLO

GIT CHECKOUT <RAMA>

10 AÑADE SUS CORRECCIONES (OPCIONAL)

GIT ADD .

GIT COMMIT -M <MENSAJE>

GIT PUSH -U ORIGIN <RAMA>

11 UNA VEZ QUE TODOS DAN EL VISTO BUENO

ACTUALIZAMOS LA VERSIÓN EN LOS FICHEROS MAVEN
Y HACEMOS EL ADD, COMMIT Y PUSH EN LA RAMA DE
DESARROLLO

PASOS CICLO COMPLETO

12 APROBAMOS EL PULL REQUEST VÍA GITHUB

SI LO HACEMOS A MANO SERÍA

GIT CHECKOUT MASTER

GIT MERGE <RAMA>

GIT PULL -U ORIGIN MASTER

CON ESTO SE INTEGRAN LOS CAMBIOS EN MASTER

13 GENERAMOS LA VERSIÓN

GIT TAG <VERSION>

14 CREAMOS LA VERSIÓN EN REMOTO

GIT PULL --TAGS

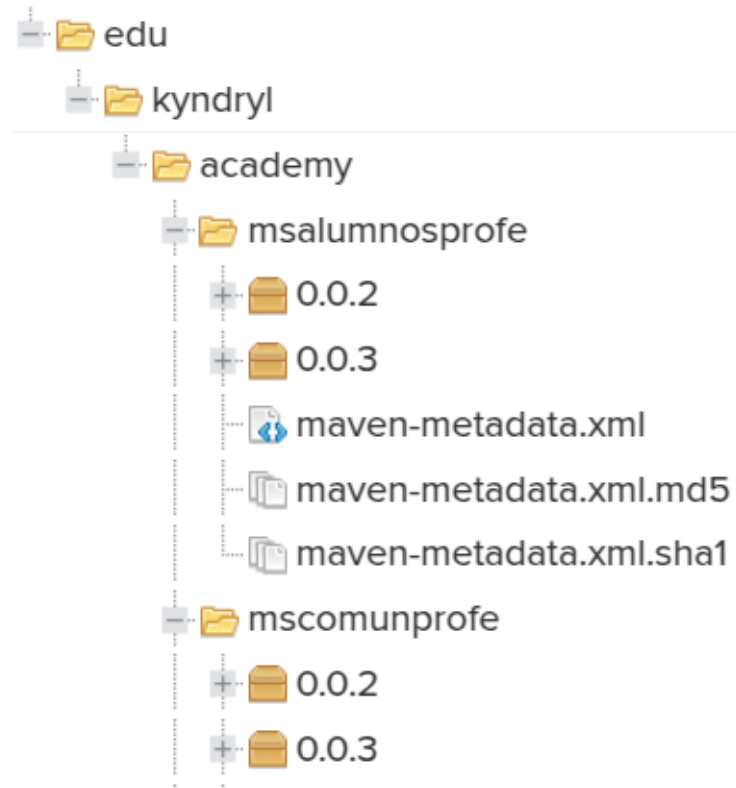
APUNTES EXTRA

CON LA CONFIGURACIÓN DE NUESTRO JENKINS,
SALTA LA EJECUCIÓN AL DETECTAR UNA NUEVA
VERSIÓN PÚBLICA

DEBIDAMENTE CONFIGURADO NEXUS, LOS NUEVOS
MICORSERVICIOS SE DESPLEGAN EN EL
REPOSITORIO CORPORATIVO

APUNTES EXTRA

**ACCEDEMOS AL REPOSITORIO PÚBLICO EN NEXUS Y
DEBERÍAMOS VER GENERADA LA NUEVA VERSIÓN**





DOCKER COMPOSE

DOCKER COMPOSE ES UNA
HERRAMIENTA QUE NOS
PERMITE DESPLEGAR
VARIAS IMÁGENES DE
DOCKER SIMULTÁNEAMENTE

A TRAVÉS DE UN FICHERO
YML, SE INSTANCIAN Y
CONFIGURAN LAS
IMÁGENES A COORDINAR

D O K C E R C O M P O S E

**D O C K E R C O M P O S E E S U N P L U G I N D E D O C K E R Q U E
S E I N S T A L A A P A R T E M E D I A N T E E L C O M A N D O**

```
$ sudo apt-get update  
$ sudo apt-get install docker-compose-plugin
```

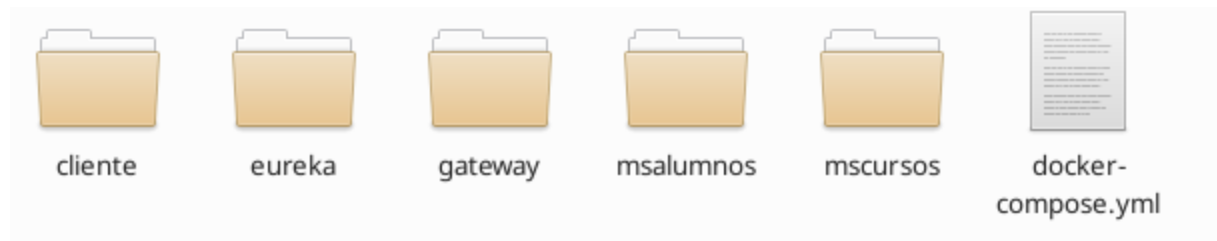
D O K C E R C O M P O S E

**CON COMPOSE VOY A PODER CREAR VARIAS
IMÁGENES Y CORRERLAS EN SU CONTENEDOR
SIMULTÁNEAMENTE**

PARA ELLO, DEBO TENER UN FICHERO YML, EN EL
DIRECTORIO RAÍZ, QUE REFERENCIA A CADA UNA DE
LAS IMÁGENES (UNA POR MICROSERVICIO) EN SU
RESPECTIVA SUBCARPETA

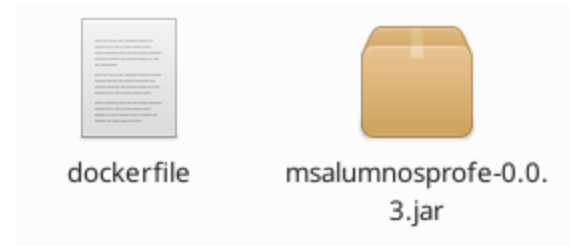
D O K C E R C O M P O S E

LA ESTRUCTURA QUEDARÍA ASÍ



D O K C E R C O M P O S E

CADA SUBCARPETA CONTIENE SU DOCKERFILE Y EL JAR FINAL CORRESPONDIENTE



```
from eclipse-temurin:17.0.14_7-jre-ubi9-minimal
add msalumnosprofe-0.0.3.jar alumnos.jar
entrypoint java -jar -Dspring.profiles.active=prod alumnos.jar
```

INCLUIMOS EL PERFIL DE PRODUCCIÓN COMO PARÁMETRO

D O K C E R C O M P O S E

EN EL FICHERO DE YML, REFERENCIAMOS A CADA IMAGEN ASÍ

```
docker-compose.yml x
services:
  eureka:
    build: ./eureka
    ports:
      - 8761:8761
  gateway:
    build: ./gateway
    ports:
      - 9090:9090
    depends_on:
      - eureka
    environment:
      EUREKA_HOST: 172.17.0.1
```

DOCKER COMPOSE

docker-compose.yml

×

```
services:
  eureka:
    build: ./eureka
    ports:
      - 8761:8761
  gateway:
    build: ./gateway
    ports:
      - 9090:9090
    depends_on:
      - eureka
    environment:
      EUREKA_HOST: 172.17.0.1
```

SERVICE: ENUMERA CADA IMAGEN
CON EL NOMBRE QUE QUERAMOS

PORTS: HACEMOS CORRESPONDER
EL PUERTO PÚBLICO DE LA MÁQUINA
CON EL PUERTO INTERNO DE DOCKER

HOST_PORT:CONTAINER_PORT

D O K C E R C O M P O S E

docker-compose.yml

×

```
services:
  eureka:
    build: ./eureka
    ports:
      - 8761:8761
  gateway:
    build: ./gateway
    ports:
      - 9090:9090
    depends_on:
      - eureka
    environment:
      EUREKA_HOST: 172.17.0.1
```

DEPENDS_ON: PODEMOS INDICAR
QUE ARRANQUE UN CONTENEDOR
ANTES QUE OTRO

ENVIROMENT: INCLUIMOS
PARÁMETROS QUE SERÁN USADA
EN LOS FICHEROS DE PROPIEDADES

DOCKER COMPOSE LOCALHOST

**NUESTRA BASE DE DATOS, ESTÁ FUERA DE DOCKER.
POR TANTO, DENTRO DE UN CONTENEDOR, LA IP DE
NUESTRA MÁQUINA HOST ES PARA DOCKER
HOST.DOCKER.INTERNAL.**

ESA SERÁ LA IP Y NO LOCALHOST, COMO VENÍAMOS
USANDO, LO QUE USEMOS EN LA CADENA DE
CONEXIÓN DEFINIDA EN LAS *PROPERTIES*

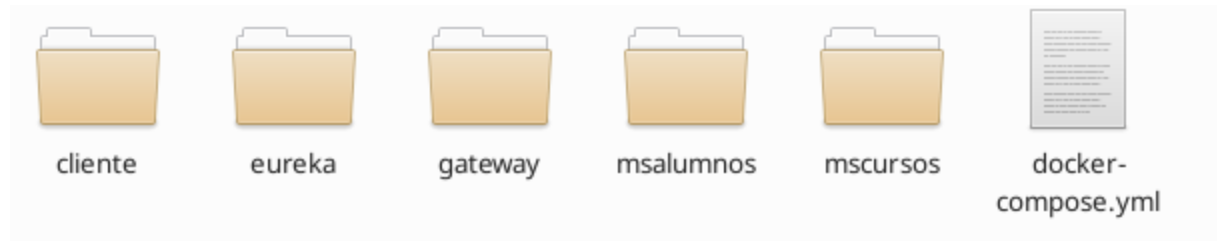
DOKCER COMPOSE LOCALHOST

SIN EMBARGO, SI ANTES USABA LOCALHOST PARA REFERIRME DESDE UN MICROSERVICIO A OTRO, AHORA AL ESTAR EN EL CONTEXTO VIRTUAL, YA NO VALE. DEBO CONSULTAR CON IFCONFIG LA IP QUE ASIGNA EL SISTEMA A DOCKER Y USAR ESA INTERFAZ

USAREMOS ESA IP PARA CONFIGURAR EUREKA O CUANDO COMUNIQUEMOS ENTRE MICROSERVICIOS DIRECTAMENTE

D O K C E R C O M P O S E

LA ESTRUCTURA QUEDARÍA ASÍ



AHORA, EJECUTO DOCKER COMPOSE UP DESDE ESTE DIRECTORIO Y TODOS LAS IMÁGENES SE RECREAN Y SE LANZAN SUS CONTEDORES

DOCKER COMPOSE

**PODEMOS CONSULTAR EL RESTO DE COMANDOS DE
COMPOSE EN LA REFERENCIA OFICIAL**

[DOCKER COMPOSE | DOCKER DOCS](#)